Architectures for Wireless Sensor Networks

Stefan Dulman [1], Paul Havinga [2]
[1] *Ambient Systems BV, the Netherlands*
stefan@ambient-systems.net
[2] *University of Twente, the Netherlands*
p.j.m.havinga@ewi.utwente.nl

Abstract

Various architectures have been developed for wireless sensor networks. Many of them leave to the programmer important concepts as the way in which the inter-task communication and dynamic reconfigurations are addressed. In this paper we describe the characteristics of a new architecture we proposed - the data-centric architecture. This architecture offers an easy way of structuring the applications designed for wireless sensor nodes that confers them superior performances.

1. Introduction

Although the nodes of a wireless sensor networks might be equipped with a power supply or energy scavenging means and an embedded processor that makes them autonomous and self-aware, their functionality and capabilities will be very limited. Therefore, collaboration between nodes is essential to deliver smart services in a ubiquitous setting.

New algorithms for networking and distributed collaboration need to be developed. These algorithms will be key for building self-organizing and collaborative sensor networks that show emergent behavior and can operate in a challenging environment where nodes move, fail, and energy is a scarce resource.

The question that rises is how to organize the internal software and hardware components in a manner that will allow them to work properly and be able to adapt dynamically to new environments, requirements and applications. At the same time the solution should be general enough to be suited for as many applications as possible. Architecture definition also includes, at the higher level, a global view of the whole network. The topology, placement of base stations, beacons, etc. are also of interest.

In this paper we present a brief state of the art of existing architectures for wireless sensor networks and introduce the concepts of a new, data-centric architecture. The new architecture offers, for a small overhead, features hard to emulate on the other existing architectures - basically, it offers dynamic on-line reconfiguration as a solution to many existing problems in sensor networks. The architecture was shown to be a working concept by the various operating systems [8], [7] and the simulator designed on top of it.

This paper is organized as follows: Section 2 looks into the characteristics of the environment in which sensor networks are deployed. Then, Section 3 presents the usual characteristics of a regular node in a sensor network. We continue discussing in Section 4 the characteristics of already existing topologies in sensor networks. Next, we present the underlying concepts of the data-centric architecture (Section 5) and we end the paper with Section 6 - the conclusions.

2. Diversity and dynamics

The environment in which sensor nodes operate can be described as *dynamic*.

As we already suggested, there may be several kinds of sensor nodes present inside a single sensor network. We could talk of heterogeneous sensor nodes from the point of view of hardware and software. From the point of view of hardware, it seems reasonable to assume that the number of a certain kind of devices will be in an inversely proportional relationship to the capabilities offered. We can assist to a tiered architecture design, where the resource poor nodes will ask more powerful or specialized nodes to make more accurate measurements of a certain detected phenomenon, to perform resource intensive operations or even to help in transmitting data at a higher distance.

Diversity can also refer to sensing several parameters and then combine them in a single decision, or in other words to perform data-fusion. We are talking about assembling together information from different kinds of sensors like: light, temperature, sound, smoke, etc. to detect for example that a fire has started.

Sensor nodes will be deployed in the real world, most probably in harsh environments. This puts them in contact with an environment that is dynamic in many senses and has a big influence on the algorithms that the sensor nodes should execute. First of all, the nodes will be deployed in a random fashion in the environment and in some cases, some of them will be mobile. Secondly, the nodes will be subject to failures at random times and they will also be allowed to change their transmission range to better suit their energy budget. This leads to the full picture of a network topology in a continuous change. The algorithms for the wireless sensor networks have as one of their characteristic the fact that they do not require a predefined well-known topology.

One more consequence of the real world deployment is that there will be many factors influencing the sensors in contact with the phenomenon. Individual calibration of each sensor node will not be feasible and probably will not help much as the external conditions will be in a continuous change. The

0-7803-9399-6/05/$20.00 © 2005 IEEE

ISSNIP 2005

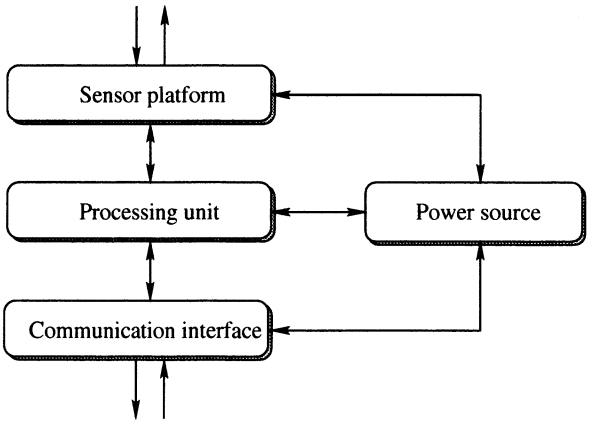

Fig. 1: Sensor node components

sensor network will calibrate itself as a reply to the changes in the environment conditions. More than this, the network will be capable of self-configuration and self-maintenance.

Another issue we need to talk about is the dynamic nature of the wireless communication medium. Wireless links between nodes can periodically appear or disappear due to the particular position of each node. Bidirectional links will coexist with unidirectional ones and this is a fact that the algorithms for wireless sensor networks need to consider.

3. Sensor Node Architecture

Current existing technology already allows integration of functionality for information gathering, processing and communication in a tight packaging or even in a single chip. The four basic blocks needed to construct a sensor node are (see Figure 1):

- *Sensor platform*

  The sensors are the interfaces to the real world. They collect the necessary information and have to be monitored by the central processing unit. The platforms may be built in a modular way such that a variety of sensors can be used in the same network. The utilization of a very wide range of sensors (monitoring characteristics of the environment such as light, temperature, air pollution, pressure, etc.) is envisioned. The sensing unit can also be extended to contain one or more actuation units (e.g. to give the node the possibility to re-position itself).
- *Processing unit*

  The processing unit is the central intelligence of the sensor node. It will not only collect the information detected by the sensor but will also be used in the communication with the network. The level of intelligence in the sensor node will strongly depend on the type of information that is gathered by its sensors and by the way in which the network operates. The sensed information will be pre-processed to reduce the amount of data to be transmitted via the wireless interface. The processing unit will also have to execute some networking protocols in order to forward the results of the sensing operation through the network to the requesting user.
- *Communication interface*

  The communication interface is the link of each node to the sensor network itself. The focus relies on a wireless communication link, in particular on the radio communication, although visible or infrared light, ultrasound, etc. means of communications have already been used [3]. The used radio transceivers can usually function in simplex mode only, and can be completely turned off, in order to save energy.
- *Power source*

  Due to the application areas of the sensor networks, autonomy is an important issue. Sensor nodes are usually equipped with a power supply in the form of one or more batteries. Current studies focus on reducing the energy consumption by using low power hardware components and advanced networking and data management algorithms. The usage of such energy scavenging techniques for sensor nodes might make possible for the sensor nodes to be self-powered. No matter which form of power source is used, energy is still a scarce resource and a series of trade-offs will be employed during the design phase to minimize its usage.

Sensor networks will be heterogeneous from the point of view of the types of nodes deployed. Moreover, whether or not a any specific sensor node can be considered as being part of the network only depends on the correct usage and participation in the sensor network suite of protocols and not on the node's specific way of implementing software or hardware. An intuitive description given in [5] envisions a sea of sensor nodes, some of them being mobile and some of them being static, occasionally containing tiny isles of relatively resource-rich devices. Some nodes in the system may execute autonomously (e.g. forming the backbone of the network by executing network and system services, controlling various information retrieval and dissemination functions, etc.), while others will have less functionality (e.g. just gathering data and relaying it to a more powerful node).

Thus, from the sensor node architecture point of view we can distinguish between several kinds of sensor nodes. A simple yet sufficient in the majority of the cases approach would be to have two kind of nodes: *high-end sensor nodes* (nodes that have plenty of resources or superior capabilities; the best candidate for such a node would probably be a fully equipped PDA device or even a laptop) and *low-end nodes* (nodes that have only the basic functionality of the system and have very limited processing capabilities).

The architecture of a sensor node consists of two main components: defining the needed functionality and how to join various elements to form a coherent sensor node. In other words, sensor node architecture means defining the exact way in which the selected hardware and software components connect to each other, how they communicate and how they interact with the central processing unit, etc.

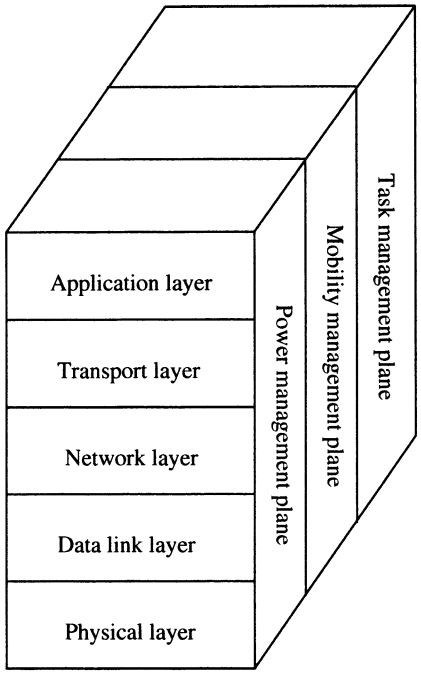

Fig. 2: Protocol stack representation of the architecture [1]

A large variety of sensor node architectures have been built up to this moment. As a general design rule, all of them have targeted the following three objectives: energy efficiency, small size and low cost. Energy efficiency is by far the most important design constraint because energy consumption depends on the lifetime of the sensor nodes. As the typical scenario of sensor networks deployment assumes that the power supplies of nodes will be limited and not rechargeable, a series of trade-offs need to be made to decrease the amount of consumed energy. Small size of the nodes leads to the ability of deploying lots of them to study a certain phenomenon. The ideal size is suggested by the name of one of the first research projects in the area: Smart Dust [11]. Very cheap sensor nodes will lead to rapid deployment of such networks and large scale usage.

4. Sensor Network Architectures

A sensor network is a very powerful tool when compared to a single sensing device. It consists of a large number of nodes, equipped with a variety of sensors that are able to monitor different characteristics of a phenomenon. A dense network of such small devices, will give the researcher the opportunity to have a spatial view over the phenomenon and, at the same time, it will produce results based on a combination of various sorts of sensed data.

Each sensor node will have two basic operation modes: initialization phase and operation phase. But, the network as a whole will function in a smooth way, with the majority of the nodes in the operation mode and only a subset of nodes in the initialization phase. The two modes of operation for the sensor nodes have the following characteristics:

- *Initialization mode*

  A node can be considered in initialization mode if it tries to integrate itself in the network and is not performing its routine function. A node can be in initialization mode for example at power on or when it detects a change in the environment and needs to configure itself. During initialization, the node can pass through different phases such as detecting its neighbors and the network topology, synchronizing with its neighbors, determining its own position or even performing configuration operations on its own hardware and software. At a higher abstraction level, a node can be considered in initialization mode if it tries to determine which services are already present in the network, which services it needs to provide or can use.

- *Operation mode*

  After the initialization phase the node enters a stable state, the regular operation state. It will function based on the conditions determined in the initialization phase. The node can exit the operation mode and pass through an initialization mode if either the physical conditions around it or the conditions related to the network or to itself have changed. The operation mode is characterized by small bursts of node activity (such as reading sensors values, performing computations or participating in networking protocols) and periods spent in an energy-saving low power mode.

A. Protocol stack approach

A first approach to building a wireless sensor network will be to use a layered protocol stack as a starting point, as in the case of traditional computer network.

The main difference between the two kinds of networks is that the blocks needed to build the sensor network usually span themselves over multiple layers, while depending on each-others. This characteristic of sensor networks comes from the fact that they have to provide functionality that is not present in traditional networks. Figure 3 presents an approximate mapping of the main blocks onto the traditional OSI protocol layers [5].

The authors of [1] propose an architecture based on the five OSI layers together with three management planes that go throughout the whole protocol stack (see Figure 2). A brief description of the layers included can be: the physical layer addresses mainly the hardware details of the wireless communication mechanism: the modulation type, the transmission and receiving techniques, etc. The data link layer is concerned with the Media Access Control (MAC) protocol that manages communication over the noisy shared channel. Routing the data between the nodes is managed by the network layer, while the transport layer helps to maintain the data flow. Finally, the application layer contains (very often) only one single user application.

In addition to the five network layers, the three management planes have the following functionality: the power management plane coordinates the energy consumption inside the sensor node. It can, for example, based on the available amount

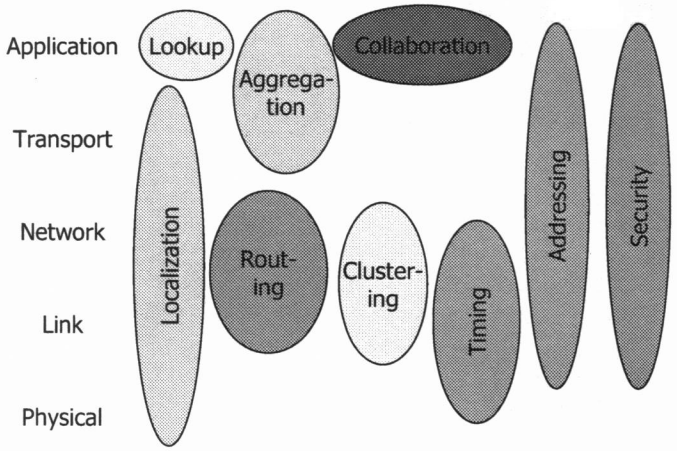

Fig. 3: Relationship building blocks / OSI layers

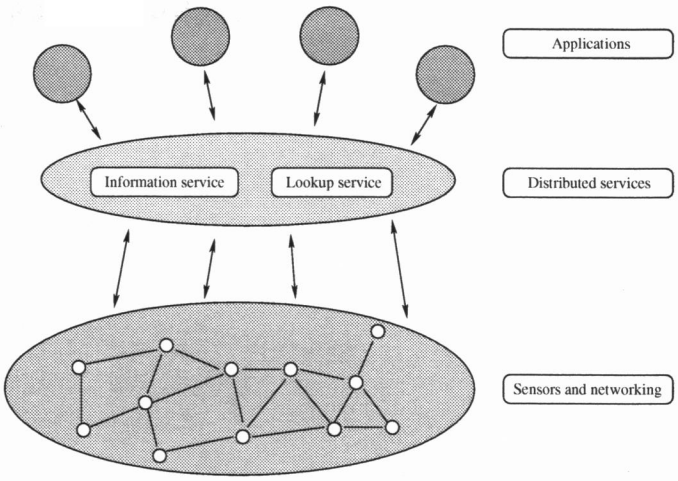

Fig. 4: EYES project architecture description

of energy, allow the node to take part in certain distributed algorithms or to control the amount of traffic it wants to forward. The mobility management plane will manage all the information regarding the physical neighbors and their movement patterns as well as its own moving pattern. The task management plane coordinates sensing in a certain region based on the number of nodes and their placement (in very densely deployed sensor networks, energy might be saved by turning certain sensors off to reduce the amount of redundant information sensed).

B. EYES project approach

The approach taken in the EYES project [9] consists of only two key system abstraction layers: the sensor and networking layer and the distributed services layer (see Figure 4). Each layer provides services that may be dynamically specified and reconfigured.

- **Sensors and networking layer** - This layer contains the sensor nodes (the physical sensor and wireless transmission modules) and the network protocols. Ad-hoc routing protocols allow messages to be forwarded through multiple sensor nodes taking into account the mobility of nodes and the dynamic change of topology. Communication protocols must be energy-efficient since sensor nodes have very limited energy supplies. To provide more efficient dissemination of data, some sensors may process data streams, and provide replication and caching.
- **Distributed services layer** - This layer contains distributed services for supporting mobile sensor applications. Distributed services coordinate with each other to perform decentralized services. These distributed servers may be replicated for higher availability, efficiency and robustness. We have identified two major services. The look-up service supports mobility, instantiation, and reconfiguration. The information service deals with aspects of collecting data. This service allows vast quantities of data to be easily and reliably accessed, manipulated, disseminated, and used in a customized fashion by applications.

On top of this architecture applications can be built using the sensor network and distributed services. Communication in a sensor network is data-centric since the identity of the numerous sensor nodes is not important, only the sensed data (together with the time and the location information) counts. The three main functions of the nodes within a sensor network are directly related to this:

- **Data discovery** - Data will be collected using the several classes of sensors employed in the network. Specialized sensors can monitor climatic parameters (humidity, temperature, etc.), motion detection, vision sensors and so on. A first step of data pre-processing can also be included in this task.
- **Data processing and aggregation** - This task is directly related to performing distributed computations on the sensed data and also aggregating several observations into a single one. The goal of this operation is the reduction of energy consumption. Data processing influences it by the fact that the transmission of one (raw sensed) data packet is equivalent to many thousands of computation cycles in the current architectures. Data aggregation keeps the overall traffic low by inspecting the contents of the routed packets, and in general, reducing the redundancy of the data in traffic by combining several similar packets into a single one.
- **Data dissemination** - This task includes the networking functionality comprising routing, multicasting, broadcasting, addressing, etc.

The existing network scenarios contain both static and mobile nodes. In some cases, the static nodes can be considered to form a back-bone of the network and are more likely to be preferred in certain distributed protocols. Both mobile and static nodes will have to perform data dissemination, so the protocols should be designed to be invariant to node mobility. The particular hardware capabilities of each kind of sensor node will determine how the previously described tasks will be mapped onto them (in principle all the nodes could provide all the previous functionality). During the initialization phase

of the network, the functionality of every node will be decided based on both the hardware configurations and the particular environmental conditions.

For a large sensor network to be able to function correctly, a tiered architecture is needed [4]. This means that nodes will have to organize themselves into clusters based on certain conditions. The nodes in each cluster will elect a leader - the best fitted node to perform coordination inside the cluster (this can be for example the node with the highest amount of energy, or the node having the most advanced hardware architecture, or just a random node). The cluster leader will be responsible for scheduling the node operations, managing the resources and the cluster structure and maintaining communication with the other clusters.

We can talk about several types of clusters that can coexist in a single network:

- **Geographical clustering** - The basic mode of organizing the sensor network. The clusters are built based on the geographical proximity. Neighboring nodes (nodes that are in transmission range of each-other) will organize themselves into groups. This operation can be handled in a completely distributed manner and it is a necessity for the networking protocols to work even when the network scales up.
- **Information clustering** - The sensor nodes can be grouped into information clusters based on the services they can provide. This clustering structure belongs to the distributed services layer and is built on top of the geographical clustering. Nodes using this clustering scheme need not be direct neighbors from the physical point of view.
- **Security clustering** - An even higher order hierarchy appears if security is taken into consideration. Nodes can be grouped based on their trust levels or based on the actions they are allowed to perform or resources they are allowed to use in the network.

Besides offering increased capabilities to the sensor network, clustering is considered one of the principal building blocks for the sensor networks also from the point of view of energy consumption. The overhead given by the energy spent for creating and organizing the sensor network is easily recovered in the long term due to the reduced traffic it leads to.

5. Data centric architecture

As we previously stated, the layered protocol stack description of the system architecture for a sensing node cannot cover all the aspects involved (such as cross-layer communication, dynamic update, etc.). In this section we address the problem of describing the system architecture in a more suited way and its implications in the application design.

A. Motivation

The sensor networks are dynamic from many points of view. Continuously changing behaviors can be noticed in several aspects of sensor networks, some of them being:

- **Sensing process** - The natural environment is dynamic by all means (the basic purpose of sensor networks is to detect, measure and alert the user of the changing of its environment). The sensor modules themselves can become less accurate, need calibration or even break down.
- **Network topology** - One of the features of the sensor networks is their continuously changing topology. There are a lot of factors contributing to this, such as: failures of nodes or the unreliable communication channel, mobility of the nodes, variations of the transmission ranges, clusters reconfiguration, addition/removal of sensor nodes, etc. Related to this aspect, the algorithms designed for sensor networks need to have two main characteristics: they need to be independent on the network topology and need to scale well with the network size.
- **Available services** - Mobility of nodes, failures or availability of certain kinds of nodes might trigger reconfigurations inside the sensor network. The functionality of nodes may depend on existing services at certain moments and when they are no longer available, the nodes will either reconfigure themselves or try to provide these services themselves.
- **Network structure** - New kinds of nodes may be added to the network. Their different and increased capabilities will bring changes to the regular way in which the network functions. Software modules might be improved or completely new software functionality might be implemented and deployed in the sensor nodes.

Most wireless sensor network architectures currently use a fixed layered structure for the protocol stack in each node. This approach has certain disadvantages for wireless sensor networks. Some of them are:

- **Dynamic environment** - Sensor nodes address a dynamic environment where nodes have to reconfigure themselves to adapt to the changes. Since resources are very limited, reconfiguration is also needed in order to establish an efficient system (a totally new functionality might have to be used if energy levels drop under certain values). The network can adapt its functionality to a new situation, in order to lower the use of the scarce energy and memory resources, while maintaining the integrity of its operation.
- **Error control** - Error control normally resides in all protocol layers so that for all layers the worst case scenario is covered. For a wireless sensor network this redundancy might be too expensive. Adopting a central view on how error control is performed and cross-layer design will reduce the resources spent for error control.
- **Power control** - Power control is traditionally done only at the physical layer, but since energy consumption in sensor nodes is a major design constraint, it is found in all layers (physical, data-link, network, transport and application layer).
- **Protocol place in the sensor node architecture** - An issue arises when trying to place certain layers in the

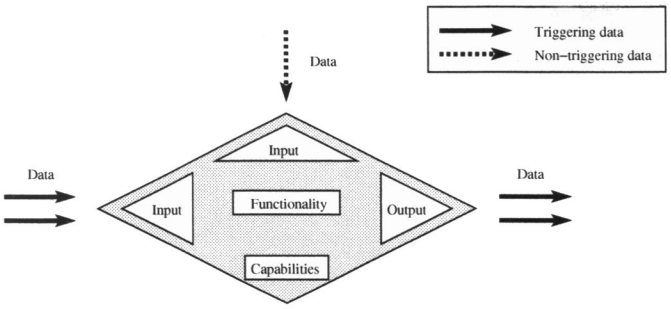

Fig. 5: Entity description

protocol stack. Examples may include: timing and synchronization, localization and calibration. These protocols might shift their place in the protocol stack as soon as their transient phase is over. The data produced by some of these algorithms might make a different protocol stack more suited for the sensor node (e.g. a localization algorithm for static sensor networks might enable a better routing algorithm that uses information about the location of the routed data destination).

- **Protocol availability** - New protocols might become available after the network deployment or at certain moments, in specific conditions, some of the sensor nodes might use a different protocol stack that better suits their goal and the environment.

It is clear from these examples that dynamic reconfiguration of each protocol as well as dynamic reconfiguration of the active protocol stack is needed.

B. Architecture description

The system we are trying to model is an event-driven system, meaning that it reacts and processes the incoming events and afterward, in the absence of these stimuli, it spends its time in the sleep state (the software components running inside the sensor node are not allowed to perform blocking waiting).

Let us name a higher level of abstraction for the event class as *data*. Data may encapsulate the information provided by one or more events, have a unique name and contain additional information such as deadlines, identity of producer, etc. Data will be the means used by the internal mechanisms of the architecture to exchange information components.

In the following we will address any protocol or algorithm that can run inside a sensor node with the term *entity* (see Figure 5). An entity is a software component that will be triggered by the availability of one or more data types. While running, each entity is allowed to read available data types (but not wait for additional data types becoming available). As a result of the processing, each software component can produce one or more types of data (usually on their exit).

An entity is also characterized by some *functionality*, meaning the sort of operation it can produce on the input data. Based on their functionality, the entities can be classified as being part of a certain protocol layer as in the previous description. For one given functionality, several entities might exist inside

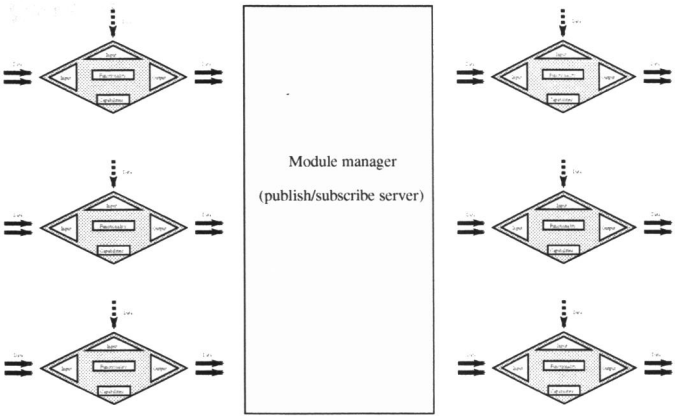

Fig. 6: Data-centric architecture

a sensor node; to discern among them, one should take into consideration their *capabilities*. By capability we understand high-level description containing the cost for a specific entity to perform its functionality (as energy, resources, time, etc.) and some characteristics indicating the estimated performance and quality of the algorithm.

In order for a set of components to work together, the way in which they have to be interconnected should be specified. The existent architectures in the wireless sensor network field, assume a fixed way in which these components can be connected, which is defined at compile time (except for the architectures that for example allow execution of agents). To change the protocol stack in such an architecture, the user should download the whole compiled code into the sensor node (via the wireless interface) and then make use of some boot code to replace the old running code in it. In the proposed architecture we are allowing this interconnection to be changed at run time, thus making on-line update of the code possible, the selection of a more suited entity to perform some functionality based on the changes in the environment, etc. (in one word allowing the architecture to become dynamically reconfigurable).

To make this mechanism work, a new entity needs to be implemented; we call this the *data manager*. The data manager will monitor the different kinds of data being available and will coordinate the data flow inside the sensor node. At the same time it will select the most fitted entities to perform the work and it will even be allowed to change the whole functionality of the sensor node based on the available entities and external environment (see Figure 7).

The implementation of these concepts can not make an abstraction of the small amount of resources each sensor node has (as energy, memory, computation power, etc.). Going down from the abstraction level to the point where the device is actually working, a compulsory step is implementing the envisioned architecture in a particular operating system (in this case maybe a better term is system software). A large range of operating systems exist for embedded systems in general [10], [13]. Scaled down versions with simple schedulers and limited functionality have been developed especially for wireless

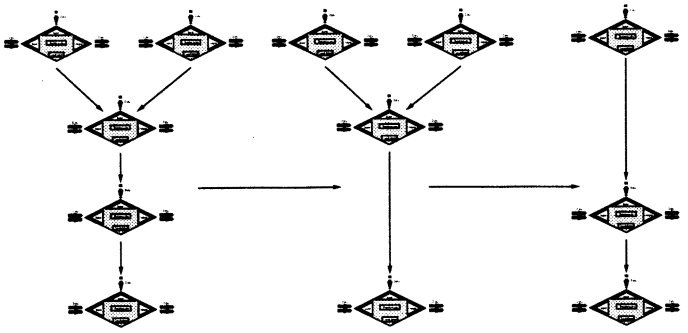

Fig. 7: Architecture transitions

sensor networks [6].

Usually, the issues of system architecture and operating system are treated separately, both of them trying to be as general as possible and to cover all the possible application cases. A simplistic view of a running operating system is a scheduler that manages the available resources and coordinates the execution of a set of tasks. This operation is centralized from the point of view of the scheduler that is allowed to take all the decisions. Our architecture can also be regarded as a centralized system, with the data manager coordinating the data flow of the other entities. To obtain the smallest overhead possible there should be a correlation between the function of the central nucleus from our architecture and the function of the scheduler from the operating system. This is why we propose a close relationship between the two concepts by extending the functionality of the scheduler with the functionality of the data manager. The main challenges that arise are keeping the size of the code low and the context-switching time.

C. Additional requirements

As we mentioned earlier, the general concept of *data* is used rather than the *event* one. For the decision based on data to work, there are some additional requirements to be met.

First of all, all the modules need to declare the name of the data that will trigger their action, the name of the data they will need to read during their action (this can generically incorporate all the shared resources in the system) and the name of the data they will produce. The scheduler needs all this information to take the decisions.

From the point of view of the operating system, a new component that takes care of all the data exchange needs to be implemented. This would in fact be an extended message passing mechanism, with the added feature of notifying the scheduler when new data types become available. The mapping of this module in the architecture is the constraint imposed to the protocols to send/receive data via, for example, a publish/subscribe mechanism to the central scheduler.

An efficient naming system for the entities and the data is needed. Downloading new entities to a sensor node involves issues similar to services discovery. Several entities with the same functionality but with different requirements and capabilities might co-exist. The data centric scheduler has to make the decision which one is the best.

D. Extension of the architecture

The architecture presented earlier might be extended to groups of sensor nodes. Several Data Centric Schedulers together with a small, fixed number of protocols can communicate with each other and form a virtual backbone of the network.

Entities running inside sensor nodes can be activated using data types that become available at other sensor nodes (for example, imagine one node using his neighbor routing entity because it needs the memory to process some other data).

Of course, this approach raises new challenges. A naming system for the functionality and data types, reliability issues of the system (for factors such as mobility, communication failures, node failures, security attacks) are just a few examples. Related work on these topics already exist (for example: [2], [12]).

E. Example

In order to better understand the flexibility offered by this architecture, we will proceed with an example. Figure 8 illustrates the case of a dynamic architecture for a sensor node.

Let us assume that the goal of the sensor network is to monitor and analyze a certain feature of the environment - as the quality of perishable merchandise stored in containers in a large warehouse. The nodes are attached to containers, so they are static for most of the time; this scenario involves a limited amount of mobility.

The nodes will start running a basic Media Access Control (MAC) protocol which makes communication possible between each other. The monitoring application will run directly on top of the MAC layer, basically storing sampled data from the actual sensors. As soon as the MAC layer has gathered enough information about the neighbors of the node, an ID based routing block can be loaded into memory to allow the dissemination of the sensed data.

In parallel, a security module could be loaded for example to determine the keys needed for each node to encrypt their data. At the same time, a link layer module could also run in order to make communication more reliable and to save energy.

As the routing protocol gathers even more data about the neighborhood, a timing and synchronization protocol can run, in order to give all the nodes the same notion of time.

A new transition towards a more efficient architecture is possible right now. As the set of needed keys for encrypting the data have been gathered, there is no need for the security module to occupy memory any longer. It can be unloaded and, for example, a localization protocol can start running. Its results will enable the employment of a geographic routing block, known for its increased performances and reduced memory usage when compared to an ID based routing scheme.

As position is determined, the localization block can be unloaded from memory as it is not useful anymore (or at least until the position of the node changes). Running an efficient

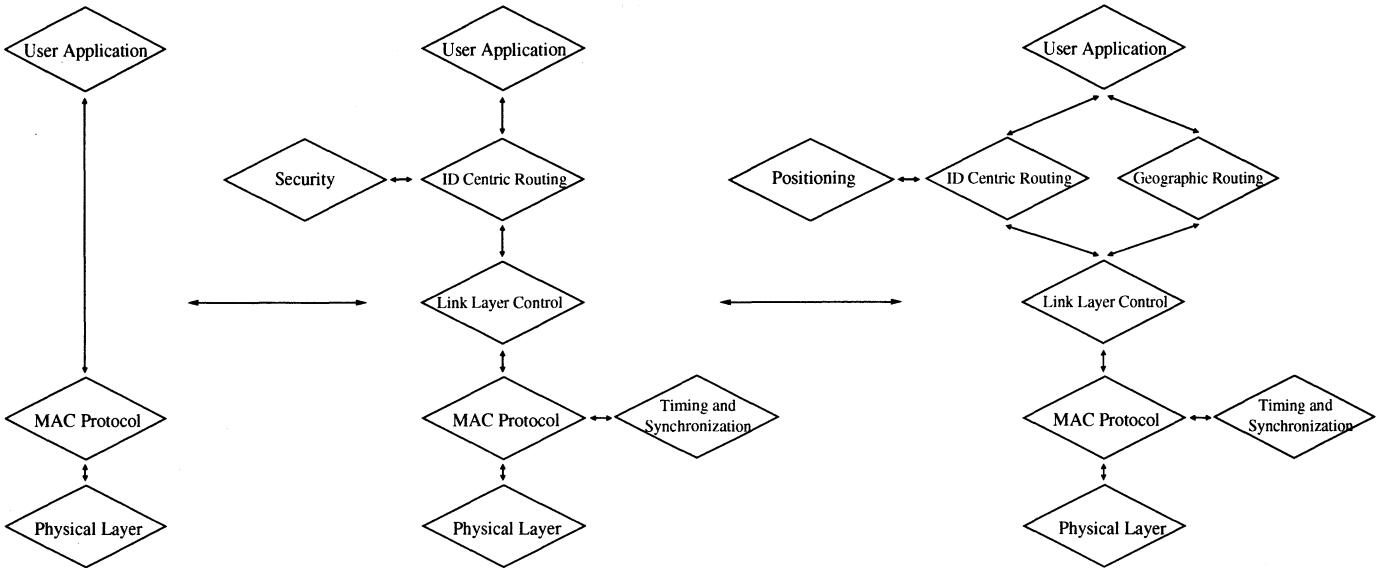

Fig. 8: Example of architecture transitions

protocol stack, having notion of time and position, being capable of encrypting data, the node is ready for continuous monitoring of the environment and has the resources free in order to load data processing algorithms or to participate in efficient data dissemination protocols.

In the case of failures, mobility, availability of new algorithms, low energy levels, etc. the architecture of the node can change again on the fly, suiting the current running situation without the need of a human operator. Loading new blocks into the memory is easy due to the publish/subscribe mechanism that does not require a fixed protocol stack and enables easy sharing of data between any number of protocols (no matter where they might be placed in the protocol stack).

6. Conclusions

In this paper we have outlined the characteristics of wireless sensor networks from an architectural point of view. As sensor networks are designed for specific applications, there is no precise architecture to fit them all but rather a common set of characteristics that can be taken as a starting point.

The combination of the data centric features of sensor networks and the need to have a dynamic reconfigurable structure has led to a new architecture that provides enhanced capabilities than the existing ones. The new architecture characteristics and implementation issues have been discussed, laying the foundations for future work.

We have briefly presented some of the existing architectures and described the ideas behind the data-centric architecture we proposed earlier.

The new architecture offers improved functionality and gives the user more flexibility in designing sensor network protocols and applications.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communication Magazine*, 40(8):102–114, August 2002.

[2] E. Cheong, J. Liebman, J. Liu, and F. Zhao. TinyGALS: a programming model for event-driven embedded systems. In *Proceedings of the 2003 ACM symposium on Applied computing*, pages 698–704. ACM Press, 2003.

[3] P.B. Chu, N.R. Lo, E. Berg, and K.S.J. Pister. Optical communication using micro corner cuber reflectors. In *MEMS97*, pages 350–355, Nagoya, Japan, January 1997.

[4] D. Estrin, R. Govindan, J.S. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Mobile Computing and Networking*, pages 263–270, 1999.

[5] P. Havinga. Eyes deliverable 1.1 - system architecture specification.

[6] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[7] T.J. Hofmeijer. The development of system software to support a data centric real-time architecture for sensor networks. Master's thesis, Twente University, july 2004.

[8] J. Mulder. Peeros preemptive eyes real-time operating system. Master's thesis, Twente University, april 2003.

[9] Eyes European Project. http://www.eyes.eu.org.

[10] Salvo. Pumpkin Incorporated, http://www.pumpkininc.com.

[11] SmartDust. http://robotics.eecs.berkeley.edu/ pister/SmartDust.

[12] P. Verissimo and A. Casimiro. Event-driven support of real-time sentient objects. In *Proceedings of WORDS 2003*, 2003.

[13] VxWorks. Wind River, http://www.windriver.com.